**PATENT**

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

| | |
|---|---|
| In re application of: | |
| Wu et al. | Group Art Unit: 2157 |
| Application No. 09/609,690 | Examiner: Gold, Avi M. |
| Filed: 07/05/2000 | Atty. Docket No. NAI1P069/99.074.01 |
| For: HIGH PERFORMANCE PACKET PROCESSING USING A GENERAL PURPOSE PROCESSOR | Date: 06/30/2008 |

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

**ATTENTION: Board of Patent Appeals and Interferences**

**REPLY BRIEF (37 C.F.R. § 41.37)**

This Reply Brief is being filed within two (2) months of the mailing of the Examiner's Answer mailed on 04/29/2008.

Following is an issue-by-issue reply to the Examiner's Answer.

Issue # 1:

The Examiner has rejected Claim 1 under 35 U.S.C. 102(e) as being anticipated by Irwin (U.S. Patent No. 6,393,026).

*Group #1: Claim 1*

With respect to Claim 1, the Examiner has relied on the following excerpts from Irwin to meet appellant's claimed "a first data processing unit adapted to filter incoming packets." Specifically, the Examiner has argued that "Irwin discloses a data packet processing system that forwards packets and assigns a program counter."

> "The data packet processing is performed by executing a packet switching software of the router. The processing for forwarding a data packet can be broken into a number of procedures. Typically, packet forwarding requires procedures, such as translation of a header of the data packet to identify suitable routes, classification of a packet flow according to source and destination addresses..." (Col. 5, lines 46-52 - emphasis added)
>
> "[serv]ice route pruning, metric route pruning, policy route pruning, option processing, congestion control, scheduling, and performance monitoring." (Col. 5, lines 53-55 - not specifically cited - emphasis added)
>
> "As each data packet enters the data packet processing system, a program counter is assigned to the packet for the duration of the time the packet is held in the data packet processing system." (Col. 6, lines 5-8)

Appellant respectfully asserts that the excerpts from Irwin relied upon by the Examiner merely disclose that "[t]he processing for forwarding a data packet can be broken into a number of procedures such as translation of a header of the data packet to identify suitable routes, classification of a packet flow according to source and destination addresses, type of service route pruning, metric route pruning, policy route pruning, option processing, congestion control, scheduling, and performance monitoring" (emphasis added). However, merely disclosing that forwarding a data packet may be broken into procedures such as the translation of a header to identify suitable routes, and classification of a packet flow according to source and destination addresses, as in Irwin, fails to suggest "a first data processing unit adapted to filter incoming packets" (emphasis added), as claimed by appellant. Clearly, procedures for forwarding a data

packet, as in Irwin, simply fails to even suggest "filter[ing] incoming packets," as claimed by appellant.

In the Examiner's Answer mailed 04/29/2008, the Examiner has argued that "[a]s seen in column 5, lines 47-65, the forwarding program determines the outgoing route to be followed by a data packet" and that "[t]he definition of filtering is a device or program that separates data, which is exactly what the forwarding of Irwin is doing." Further, the Examiner has argued that "[t]he packets are forwarded to different routes, thus being separated, which is the very definition of filter." In addition, the Examiner has argued that "[t]he data packet processing system, on column 6, lines 5-8, assigns a program counter to incoming packets and forwards them using the forwarding program; thus Irwin clearly teaches 'a first data processing unit adapted to filter incoming packets.'"

Appellant respectfully disagrees and asserts that the excerpts from Irwin relied upon by the Examiner merely teach that "[t]he processing for forwarding a data packet can be broken into a number of procedures" and that "[t]ypically, packet forwarding requires procedures, such as translation of a header of the data packet to identify suitable routes, classification of a packet flow according to source and destination addresses, type of service route pruning, metric route pruning, policy route pruning, option processing, congestion control, scheduling, and performance monitoring" (Col. 5, lines 47-55 – emphasis added). Further, the excerpts teach that "a program counter is assigned to the packet for the duration of the time the packet is held in the data packet processing system" (Col. 6, lines 5-8 – emphasis added).

However, the disclosure that packet forwarding requires procedures such as translation of a header to identify suitable routes, classification of a packet, route pruning, option processing, congestion control, scheduling, and performance monitoring, in addition to the disclosure of assigning a program counter to the packet, as in Irwin, simply fails to suggest "a first data processing unit adapted to filter incoming packets" (emphasis added), as claimed by appellant. Clearly, procedures for packet forwarding, as in Irwin, simply fail to even suggest "filter[ing] incoming packets," as claimed by appellant.

Additionally, appellant respectfully asserts that it would be unobvious to modify Irwin to "filter incoming packets" (emphasis added), as claimed by appellant. Specifically, Irwin teaches "a router having forwarding engines at network interfaces" where "[e]ach interface 12 may be provisioned with a forwarding engine 18 that processes incoming packets in order to determine which outgoing network interface 12 needs to be accessed through the interconnect structure of the switching fabric 14" (Col. 1, lines 32-40 – emphasis added). Further, Irwin teaches that "[s]ince the network processor 16 is no longer required to make routing decisions for a data packet, the packet throughput is capable of being scaled with the number of physical interfaces 12" such that "[t]he single point of congestion as formerly experienced with the centralization of the routing algorithms is replaced by multiple processors 12" (Col. 1, lines 52-57 – emphasis added).

Appellant asserts that Irwin's router with forwarding engines at each network interface simply utilizes the forwarding engines to determine which outgoing network interface needs to be accessed through the interconnect structure in order to alleviate the network processor from making routing decisions, which allows the throughput to be scaled with each interface. Clearly, a router with a forwarding engine at each interface that determines which outgoing network interface needs to be accessed through the interconnect structure, would be unable to filter incoming packets due to the congestion as formerly experienced with the centralization of the routing algorithms, as expressly noted in Irwin. Therefore, appellant asserts that it would be unobvious to include filtering, in the manner claimed by appellant, in Irwin's multiple network interface router.

Further, in the Examiner's Answer mailed 04/29/2008, the Examiner has argued that "[a]ppellant also relies on column 1, lines 32-57 of Irwin to show that the limitation is not found, but the Examiner would like to point out, regardless, that portion of Irwin refers to figure 1 which is labeled as prior art" and that "[e]ven so, it teaches forwarding through different interfaces which is clearly filtering."

Appellant respectfully disagrees and asserts that Fig. 1 of Irwin clearly "shows an example of a router having forwarding engines at network interfaces" and that "[t]he router 10 consists of a plurality of network interfaces" where "[e]ach interface 12 may be provisioned with a

forwarding engine 18 that processes incoming packets in order to determine which outgoing network interface 12 needs to be accessed through the interconnect structure of the switching fabric 14" (Col. 1, lines 32-40 – emphasis added).

Again, appellant asserts that a router with a forwarding engine at each interface that determines which outgoing network interface needs to be accessed through the interconnect structure, as in Irwin, simply fails to suggest "a first data processing unit adapted to filter incoming packets" (emphasis added), as claimed by appellant. Clearly, a forwarding engine determining which outgoing network interface needs to be accessed, as in Irwin, simply fails to even suggest "filter[ing] incoming packets," as claimed by appellant.

Further, with respect to Claim 1, the Examiner has relied on the following excerpts from Irwin to meet appellant's claimed "a second data processing unit adapted to process incoming packets according to one of said plurality of instruction sets after the filtering, based on a thread assigned to the incoming packets by the first data processing unit." Specifically, the Examiner has argued that "Irwin discloses a thread assigned to the packets that allows the forwarding program to process the packets."

> "As each data packet enters the data packet processing system, a program counter is assigned to the packet for the duration of the time the packet is held in the data packet processing system. The program counter uniquely defines a main program flow in the main forwarding program that is executed for the packet to select an output queue of the router for routing the packet. Stepping through the program counter temporarily creates a virtual CPU execution unit, or a hardware thread, for the data packet. Each thread is represented by a thread identification number which is defined by the program counter and a register file used by the process followed for packet processing." (Col. 6, lines 5-16 - emphasis added)

Appellant respectfully disagrees and asserts that the excerpt from Irwin relied upon by the Examiner merely teaches that "[a]s each data packet enters the data packet processing system, a program counter is assigned to the packet for the duration of the time the packet is held in the data packet processing system" and that "[s]tepping through the program counter temporarily creates a virtual CPU execution unit, or a hardware thread, for the data packet" (emphasis added).

However, the mere disclosure of assigning a program counter to the packet, such that stepping through the program counter creates a virtual CPU execution unit or hardware thread for the data packet, as in Irwin, fails to teach "a second data processing unit adapted to process incoming packets according to one of said plurality of instruction sets after the filtering, based on a thread assigned to the incoming packets by the first data processing unit" (emphasis added), as claimed by appellant. Clearly, stepping through the program counter to create a hardware thread for the packet, as in Irwin, simply fails to even suggest "process[ing] incoming packets according to one of said plurality of instruction sets **after the filtering**, based on a thread" (emphasis added), as claimed by appellant.

Additionally, appellant respectfully asserts that Irwin, as argued above, simply fails to suggest any sort of "filtering," as claimed by appellant, let alone in the specific context claimed by appellant. For example, appellant asserts that Irwin merely discloses that "[w]hen the end procedural call is read and the end of the main forwarding program has been reached (S32), if there is no data packet to be processed (S34), the thread is placed in the queue of idle threads since the packet processing has been completed and the packet has been forwarded to the designated route" (Col. 10, lines 56-61 – emphasis added). Further, Irwin teaches that "[i]f there is a data packet to be processed (S34), the thread is reactivated for a new packet (S02)" (Col. 10, lines 61-62).

However, disclosing that the thread is placed in the queue of idle threads after the packet processing has been completed and the packet has been forwarded to the designated route, as in Irwin, fails to suggest "a second data processing unit adapted to process incoming packets according to one of said plurality of instruction sets after the filtering, based on a thread assigned to the incoming packets by the first data processing unit" (emphasis added), as claimed. Clearly, placing the thread in the queue of idle threads after processing is completed, as in Irwin, simply fails to even suggest "process[ing] incoming packets according to one of said plurality of instruction sets after **the filtering**, **based on a thread assigned** to the incoming packets by the first data processing unit" (emphasis added), as claimed by appellant.

In the Examiner's Answer mailed 04/29/2008, the Examiner has argued that "[a]s seen in column 6, lines 5-16, a thread is assigned to the packet by the data packet processing system, which is

the first data processing unit of the claim." Further, the Examiner has argued that "[t]he second data processing unit is in the process followed after the packet process" and that "[t]his second data processing unit uses the thread which is represented by a thread identification number." In addition, the Examiner has argued that "[t]hus, column 6, lines 5-16, clearly teaches the third limitation of claim 1."

Appellant respectfully disagrees and asserts that Irwin teaches "[a] data packet processing system for a router in accordance with the present invention uses a multiprocessor architecture where a sequence of operations is required to be executed to determine a route which an incoming data packet needs to follow in order to reach its destination" (Col. 5, lines 34-38, not specifically cited – emphasis added).

However, a data packet processing system for a router that determines a route which an incoming data packet needs to follow in order to reach its destination, as in Irwin, simply fails to even suggest "process[ing] incoming packets according to one of said plurality of instruction sets after **the filtering**" (emphasis added), as claimed by appellant. Clearly, the data packet processing system that determines a route for an incoming data packet, as in Irwin, simply fails to even suggest "process[ing] incoming packets according to one of said plurality of instruction sets **after the filtering**" (emphasis added), as claimed by appellant.

Further, the excerpt from Irwin relied upon by the Examiner teaches that "[a]s each data packet enters the data packet processing system, a program counter is assigned to the packet for the duration of the time the packet is held in the data packet processing system" and that "[s]tepping through the program counter temporarily creates a virtual CPU execution unit, or a hardware thread, for the data packet," where "[e]ach thread is represented by a thread identification number which is defined by the program counter and a register file used by the process followed for packet processing" (Col. 6, lines 5-16 – emphasis added). Additionally, Irwin teaches that in "[s]tepping through the program counter, an output queue of the router is selected for routing the packet" and that "[w]hen packet processing has been completed, the packet has been forwarded to the selected output queue and the thread is made idle or destroyed" (Col. 6, lines 29-33 – emphasis added).

However, assigning a program counter to the packet, stepping through the program counter to temporarily create a hardware thread for the data packet, where each thread is represented by a thread identification number defined by the program counter and a register file used by the process for packet processing, stepping through the program counter to select an output queue for routing the packet, and idling or destroying the thread **after the packet has been forwarded** to the selected output queue, as in Irwin, simply fails to suggest "process[ing] incoming packets according to one of said plurality of instruction sets **after the filtering**, based on a thread assigned to the incoming packets by the first data processing unit" (emphasis added), as claimed by appellant. Clearly, stepping through the program counter to temporarily create a hardware thread for the data packet, stepping through the program counter to select an output queue for routing the packet, and idling or destroying the thread **after the packet has been forwarded**, as in Irwin, simply fails to even suggest "**process[ing] incoming packets** according to one of said plurality of instruction sets **after the filtering**, based on a thread assigned to the incoming packets" (emphasis added), as claimed by appellant.

In addition, with respect to Claim 1, the Examiner has relied on the following excerpts from Irwin to meet appellant's claimed "a data bus connecting the addressable memory unit and the first and second data processing units." Specifically, the Examiner has argued that "Irwin discloses a processor bus."

> "The received requests are forwarded to a processing unit (not shown) via a processor bus 119 using a node dependent logic 132 for processing the requests." (Col. 8, lines 49-52 - emphasis added)
>
> "FIG. 8 illustrates the P1394.2 functional blocks of a transmission interface that may be used in a computing node for a multiprocessor array used in the present invention. The computing array has an east to west macro 118 and a west to east macro 120 which are connected by a processor bus 119 to a processing unit (not shown). The terms "east" and "west" are used for convenience to indicate two opposite directions in a ring topology." (Col. 8, lines 29-36 - not specifically cited - emphasis added)

Appellant respectfully disagrees and asserts that the excerpt from Irwin relied upon by the Examiner simply teaches that "received requests are forwarded to a processing unit (not shown) via a processor bus 119" (emphasis added). Further, Irwin teaches that "[t]he computing array has an east to west macro 118 and a west to east macro 120 which are connected by a processor bus 119 to a processing unit (not shown)" (Col. 8, lines 31-34 – emphasis added). However, the

mere disclosure of a processing unit connected by a processor bus to an east to west macro and a west to east macro, as in Irwin, fails to teach "a data bus connecting the addressable memory unit and the first and second data processing units" (emphasis added), as claimed by appellant.

In the Examiner's Answer mailed 04/29/2008, the Examiner has argued that "column 8, lines 49-52... show that a data bus is explicitly shown in the reference" and that "[t]here is also a processor bus shown on column 7, lines 49-52." Further, the Examiner has argued that "it is clear from reading column 6, lines 5-16, that the addressable memory unit and first and second processing units are connected by a data bus as they all load data and data is transferred from one to another and thus must be connected by a data bus" and that "[t]he data bus is inherently necessary."

Appellant respectfully disagrees and asserts that the excerpts from Irwin relied upon by the Examiner merely teach that "[t]he received requests are forwarded to a processing unit (not shown) via a processor bus 119 using a node dependent logic 132 for processing the requests" (Col. 8, lines 49-52 – emphasis added). Further, the excerpts from Irwin teach "[a] packet switching software of the router which is executed by the data packet processing system 100" (Col. 7, lines 49-50), where "[t]he data packet processing system 100 comprises two I/O nodes 102 as master nodes, and a processor array having three computing nodes 110 as slave nodes which are connected to the I/O nodes 102 in a ring topology" (Col. 7, lines 42-45 – emphasis added). In addition, the excerpts teach that "a program counter is assigned to the packet for the duration of the time the packet is held in the data packet processing system" (Col. 6, lines 5-8 – emphasis added).

However, teaching that received requests are forwarded to a processing unit via a processor bus, and that a data packet processing system comprises two I/O nodes 102 and a processor array having three computing nodes 110 connected to the I/O nodes 102 in a ring topology, in addition to teaching that a program counter is assigned to the packet, as in Irwin, simply fails to even suggest "a data bus connecting the addressable memory unit and the first and second data processing units" (emphasis added), as claimed by appellant. Clearly, assigning a program counter to the packet, as in Irwin, simply fails to even suggest an "addressable memory unit," as claimed by appellant.

Furthermore, Fig. 7 of Irwin shows that each computing node 110 of the processing array includes Memory, CPU, and Cache. Additionally, Fig. 7 of Irwin shows that each I/O node 102 includes I/O, Memory, and CCU. Clearly, computing nodes 110 that each include memory, in addition to I/O nodes 102 that each include memory, as in Irwin, simply fails to support the Examiner's allegation that "the addressable memory unit and first and second processing units are connected by a data bus as they all load data and data is transferred from one to another and thus must be connected by a data bus" (emphasis added), as alleged by the Examiner. In addition, showing that computing nodes 110 and I/O nodes 102 each include memory, as in Irwin, simply fails to even suggest "a data bus connecting the addressable memory unit and the first and second data processing units" (emphasis added), as claimed by appellant.

Additionally, it appears that the Examiner has relied on an inherency argument regarding the above emphasized claim limitations. In view of the arguments made hereinabove, any such inherency argument has been adequately rebutted, and a notice of allowance or a specific prior art showing of such claim features, in combination with the remaining claim elements is respectfully requested. (See MPEP 2112)

Further, in response, appellant asserts that the fact that a certain result or characteristic may occur or be present in the prior art is not sufficient to establish the inherency of that result or characteristic. *In re Rijckaert*, 9 F.3d 1531, 1534, 28 USPQ2d 1955, 1957 (Fed. Cir. 1993); *In re Oelrich*, 666 F.2d 578, 581-82, 212 USPQ 323, 326 (CCPA 1981). "To establish inherency, the extrinsic evidence 'must make clear that the missing descriptive matter is necessarily present in the thing described in the reference, and that it would be so recognized by persons of ordinary skill. Inherency, however, may not be established by probabilities or possibilities. The mere fact that a certain thing may result from a given set of circumstances is not sufficient.'" *In re Robertson*, 169 F.3d 743, 745, 49 USPQ2d 1949, 1950-51 (Fed. Cir. 1999).

The Examiner is reminded that a claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described in a single prior art reference. Verdegaal Bros. v. Union Oil Co. Of California, 814 F.2d 628, 631, 2 USPQ2d 1051, 1053 (Fed. Cir. 1987). Moreover, the identical invention must be shown in as complete detail as contained

in the claim. Richardson v. Suzuki Motor Co.868 F.2d 1226, 1236, 9USPQ2d 1913, 1920 (Fed. Cir. 1989). The elements must be arranged as required by the claim.

This criterion has simply not been met by the Irwin reference, especially in view of the arguments made hereinabove.

Issue #2:

The Examiner has rejected Claims 2-16, and 30 under 35 U.S.C. 103(a) as being unpatentable over Irwin (U.S. Patent No. 6,393,026), in view of Kadambi et al. (U.S. Patent No. 6,850,521).

*Group #1: Claims 2 and 16*

Appellant respectfully asserts that such claims are not met by the prior art for the reasons argued with respect to Issue #1, Group #1.

To establish a *prima facie* case of obviousness, three basic criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations. The teaching or suggestion to make the claimed combination and the reasonable expectation of success must both be found in the prior art and not based on appellant's disclosure. *In re Vaeck*,947 F.2d 488, 20 USPQ2d 1438 (Fed.Cir.1991).

Appellant respectfully asserts that at least the third element of the *prima facie* case of obviousness has not been met, since the prior art excerpts, as relied upon by the Examiner, fail to teach or suggest all of the claim limitations, as noted above.

*Group #2: Claims 3, 5, 13 and 14*

With respect to the Claim 3, the Examiner has relied on the following excerpts to make a prior art showing of appellant's claimed "policy action table connected to said data bus and said addressable memory unit, wherein said policy action table stores at least one data processing policy."

> "FIG. 1 shows an example of a router having forwarding engines at network interfaces. The router 10 consists of a plurality of network interfaces. 12 that are interconnected through a switching fabric 14 that is controlled through a network processor 1G. Each interface 12 may be provisioned with a forwarding engine 18 that processes incoming packets in order to determine which outgoing network interface 12 needs to be accessed through the interconnect structure of the switching fabric 14. By locating the forwarding algorithms at the interface 12, the forwarding engine 18 is required to operate at the wire rate defined by the associated transmission interface. To allow for processing suitable for transmission facilities having wire speeds of OC-12, OC-48, and OC-192, a local version of the forwarding algorithms and data tables found normally in the network processor are downloaded from the network processor 16 to the forwarding engine 18 of the network interface 12. The forwarding engine 18 is optimized for forwarding speed whereas the routing table data management operates at a lower rate of changes occurring to the network configuration. Since the network processor 16 is no longer required to make routing decisions for a data packet, the packet throughput is capable of being scaled with the number of physical interfaces 12. The single point of congestion as formerly experienced with the centralization of the routing algorithms is replaced by multiple processors 12." (Irwin - Col. 1, lines 32-57 - emphasis added)

> "Referring once again to FIG. 14, after. FFP 141 applies appropriate configured filters and results are obtained from the appropriate rules table 22, logic 1411 in FFP 141 determines and takes the appropriate action. The filtering logic can discard the packet, send the packet to the CPU 52, modify the packet header or IP header, and recalculate any IP checksum fields or takes other appropriate action with respect to the headers." (Kadambi - Col. 35, lines 57-64 - emphasis added)

Appellant respectfully asserts that the excerpt from Irwin relied upon by the Examiner merely teaches that "a local version of the forwarding algorithms and data tables found normally in the network processor are downloaded from the network processor 16 to the forwarding engine 18 of the network interface 12" and that "the routing table data management operates at a lower rate of changes occurring to the network configuration" (emphasis added). However, merely teaching that data tables are downloaded from the network processor to the forwarding engine, and that the routing table data management operates at a lower rate, as in Irwin, simply fails to suggest a "policy action table," much less a "policy action table connected to said data bus and said

addressable memory unit, wherein said policy action table stores at least one data processing policy" (emphasis added), as claimed by appellant. Appellant notes that even the Examiner has admitted that "Irwin fails to teach [such] limitation."

Furthermore, appellant respectfully asserts that the excerpt form Kadambi relied upon by the Examiner simply teaches that "[t]he filtering logic can discard the packet, send the packet to the CPU 52, modify the packet header or IP header, and recalculate any IP checksum fields or takes other appropriate action with respect to the headers" (emphasis added). However, the mere disclosure of filtering logic that can discard the packet, send the packet to the CPU, modify the packet header, and recalculate any IP checksum fields, as in Kadambi, fails to teach a "policy action table," let alone a "policy action table connected to said data bus and said addressable memory unit, wherein said policy action table stores at least one data processing policy" (emphasis added), as claimed by appellant. Clearly, filtering logic, as in Kadambi, simply fails to even suggest a "policy action table [that] stores at least one data processing policy" (emphasis added), as claimed by appellant.

In the Examiner's Answer mailed 04/29/2008, the Examiner has argued that "[t]he portion of Irwin used in the rejection, column 1, lines 32-57, is there to show the use of a table for routing." Further, the Examiner has argued that "[a]ll of claim 3 is found in the 103 reference, Kadambi" and that "[i]t appears the [a]ppellant does not provide arguments other than stating that the portions relied on in Kadambi do not disclose the claim." In addition, the Examiner has argued that "column 35, lines 57-64... teac[h] a rules table that stores the filtering logic used to take appropriate action with" and that "[t]he rules table is equivalent to a policy action table as the filtering logic is a data processing policy." Additionally, the Examiner has argued that "[t]he filtering logic is a processing policy because it has policies for packet processing as evidenced by the various processing actions the filtering logic can perform." Furthermore, the Examiner has concluded that "[t]hus, [appellant's claimed technique] is clearly disclosed [by the] combination of Irwin and Kadambi."

Appellant respectfully disagrees and asserts that the arguments hereinabove clearly indicate as to how the excerpts from Irwin and Kadambi relied upon by the Examiner fail to meet appellant's claimed technique. Additionally, appellant asserts that the excerpt from Irwin relied upon by the

Examiner merely teaches that "[e]ach interface 12 may be provisioned with a forwarding engine 18 that processes incoming packets in order to determine which outgoing network interface 12 needs to be accessed through the interconnect structure of the switching fabric 14" and that "a local version of the forwarding algorithms and data tables found normally in the network processor are downloaded from the network processor 16 to the forwarding engine 18 of the network interface 12" (Col. 1, lines 33-48 – emphasis added). Further, Irwin teaches that "[f]orwarding of the IP datagram requires the router to lookup the destination address in a forwarding table to identify which interface the IP datagram is to be forwarded" (Col. 1, lines 19-22 – emphasis added).

However, merely teaching that forwarding algorithms and data tables are downloaded from the network processor to the forwarding engine, and that the router looks up the destination address in a forwarding table to identify an interface, as in Irwin, simply fails to suggest a "policy action table," much less a "policy action table connected to said data bus and said addressable memory unit, wherein said policy action table stores at least one data processing policy" (emphasis added), as claimed by appellant. Clearly, looking up a destination address in a forwarding table to identify an interface, as in Irwin, simply fails to even suggest that "said policy action table stores at least one data processing policy" (emphasis added), as claimed by appellant.

In addition, appellant asserts that the excerpt from Kadambi relied upon by the Examiner teaches that "after... FFP [(fast filtering processor)] 141 applies appropriate configured filters and results are obtained from the appropriate rules table 22, logic 1411 in FFP 141 determines and takes the appropriate action" and that "[t]he filtering logic can discard the packet, send the packet to the CPU 52, modify the packet header or IP header, and recalculate any IP checksum fields or takes other appropriate action with respect to the headers" (Col. 35, lines 57-64 – emphasis added).

However, obtaining results from the rules table after the fast filtering processor applies configured filters, in addition to the filtering logic discarding the packet, sending the packet to the CPU, modifying the packet header or IP header, and recalculating any IP checksum fields or taking other appropriate action with respect to the headers, as in Kadambi, simply fails to suggest a "policy action table," much less a "policy action table connected to said data bus and said addressable memory unit, wherein said policy action table stores at least one data processing

policy" (emphasis added), as claimed by appellant. Clearly, obtaining results from the rules table after the fast filtering processor applies configured filters, as in Irwin, simply fails to even suggest that "said policy action table stores at least one data processing policy" (emphasis added), as claimed by appellant.

Again, appellant respectfully asserts that at least the third element of the *prima facie* case of obviousness has not been met, since the prior art excerpts, as relied upon by the Examiner, fail to teach or suggest all of the claim limitations, as noted above.

*Group #3: Claim 4*

With respect to Claim 4, the Examiner has relied on the following excerpt from Kadambi to make a prior art showing of appellant's claimed "first address pointer element for identifying the location in said addressable memory unit of one of said plurality of instruction sets." Specifically, the Examiner has argued that "Kadambi discloses logic 1411 in the FFP 141 which points the instruction sets to take action."

> "Referring once again to FIG. 14, after. FFP 141 applies appropriate configured filters and results are obtained from the appropriate rules table 22, logic 1411 in FFP 141 determines and takes the appropriate action. The filtering logic can discard the packet, send the packet to the CPU 52, modify the packet header or IP header, and recalculate any IP checksum fields or takes other appropriate action with respect to the headers." (Col. 35, lines 57-64 - emphasis added)

Appellant respectfully disagrees and asserts that the excerpt from Kadambi relied upon by the Examiner teaches that "logic 1411 in FFP 141 determines and takes the appropriate action," and that "[t]he filtering logic can discard the packet, send the packet to the CPU 52, modify the packet header or IP header, and recalculate any IP checksum fields or takes other appropriate action with respect to the headers" (emphasis added). However, the mere disclosure of filtering logic that can determine and take the appropriate action such as discarding the packet, sending the packet to the CPU, modifying the packet header, and recalculating any IP checksum fields, as in Kadambi, fails to suggest a "first address pointer element," much less "a first address pointer element for identifying the location in said addressable memory unit of one of said plurality of instruction sets" (emphasis added), as claimed by appellant.

In the Examiner's Answer mailed 04/29/2008, the Examiner has stated that "[a]ppellant continues to provide no arguments and only states that the art does not disclose the claim" and that "[t]here is no explanation as to why the [a]ppellant disagrees with the Examiner's rejection." Further, the Examiner has again argued that "[r]egarding the first limitation, column 35, lines 57-64, Kadambi discloses logic 1411 in the FFP 141 which points to instruction sets to take action." In addition, the Examiner has argued that "Irwin discloses the addressable memory unit in claim 1" and that "[t]hus, it is shown that Kadambi combined with Irwin discloses these limitations."

Appellant respectfully disagrees and asserts that the arguments hereinabove clearly indicate as to how the excerpt from Kadambi relied upon by the Examiner fails to meet appellant's claimed technique. Furthermore, appellant again asserts that the excerpt from Kadambi relied upon by the Examiner teaches that "logic 1411 in FFP 141 determines and takes the appropriate action" (Col. 35, lines 57-60 – emphasis added). Clearly, the disclosure of logic 1411 determining and taking the appropriate action, as in Kadambi, simply fails to support the Examiner's allegation that "Kadambi discloses logic 1411 in the FFP 141 which points to instruction sets to take action" (emphasis added), as alleged by the Examiner. Furthermore, the mere disclosure of filtering logic that can determine and take the appropriate action, as in Kadambi, fails to suggest a "first address pointer element," much less "a first address pointer element for identifying the location in said addressable memory unit of one of said plurality of instruction sets" (emphasis added), as claimed by appellant.

Furthermore, appellant again asserts that Irwin merely teaches that "a program counter is assigned to the packet for the duration of the time the packet is held in the data packet processing system" (Col. 6, lines 5-8 – emphasis added). Clearly, assigning a program counter to the packet, as in Irwin, simply fails to suggest "a first address pointer element for identifying the location in said addressable memory unit of one of said plurality of instruction sets" (emphasis added), as claimed by appellant. Furthermore, assigning a program counter to the packet, as in Irwin, simply fails to even suggest an "addressable memory unit," as claimed by appellant.

Further, with respect to Claim 4, the Examiner has relied on the following excerpt from Kadambi to make a prior art showing of appellant's claimed "second address pointer element for

identifying the location in said addressable memory unit of a state block." Specifically, the Examiner has argued that "Kadambi discloses the FFP which is essentially a state machine."

> "FFP 141 is essentially a state machine driven programmable rules engine. The filters used by the FFP in a first embodiment are 64 (sixty-four) bytes wide, and are applied on an incoming packet. In some embodiments, a 64 byte filter mask can be used and applied to any selected 64 bytes or 512 bits of a packet. In another embodiment, however, a filter can be created by parsing selected fields of an incoming packet such that a 64 byte filter mask is created, which will be selectively applied to fields of interest of an incoming packet. In yet another embodiment, a filter can be created by applying a predetermined number of offsets to the incoming data packet 112, wherein a predetermined number of bytes immediately following each individual offset are parsed from the packet and thereafter concatenated together to form a filter value utilized in the filtration process." (Col. 31, lines 20-34 - emphasis added)

Appellant respectfully disagrees and asserts that the excerpt from Kadambi relied upon by the Examiner simply teaches that "FFP 141 is essentially a state machine driven programmable rules engine" (emphasis added). However, the mere disclosure of a state machine driven programmable rules engine, as in Kadambi, simply fails to suggest any sort of a "second address pointer element," much less "a second address pointer element for identifying the location in said addressable memory unit of a state block" (emphasis added), as claimed by appellant.

Additionally, in the Examiner's Answer mailed 04/29/2008, the Examiner has again argued that "[r]egarding the second limitation, column 31, lines 20-34, Kadambi discloses the FFP which is essentially a state machine." In addition, the Examiner has argued that "Irwin discloses the addressable memory unit in claim 1" and "[t]hus, it is shown that Kadambi combined with Irwin discloses these limitations."

Again, appellant respectfully disagrees and asserts that the excerpt from Kadambi relied upon by the Examiner simply teaches that "FFP 141 is essentially a state machine driven programmable rules engine" (Col. 31, lines 20-21 –emphasis added). However, teaching a programmable rules engine driven by a state machine, as in Kadambi, simply fails to suggest a "second address pointer element," much less "a second address pointer element for identifying the location in said addressable memory unit of a state block" (emphasis added), as claimed by appellant. Clearly, the simple teaching of a state machine, as in Kadambi, simply fails to even suggest "a second

address pointer element for identifying the location...of a state block" (emphasis added), as claimed by appellant.

Furthermore, appellant again asserts that Irwin merely teaches that "a program counter is assigned to the packet for the duration of the time the packet is held in the data packet processing system" (Col. 6, lines 5-8 – emphasis added). Clearly, assigning a program counter to the packet, as in Irwin, simply fails to suggest "a second address pointer element for identifying the location in said addressable memory unit of a state block" (emphasis added), as claimed by appellant. Additionally, assigning a program counter to the packet, as in Irwin, simply fails to even suggest an "addressable memory unit," as claimed by appellant.

Again, appellant respectfully asserts that at least the third element of the *prima facie* case of obviousness has not been met, since the prior art excerpts, as relied upon by the Examiner, fail to teach or suggest all of the claim limitations, as noted above.

*Group #4: Claims 6-12*

With respect to Claim 6, the Examiner has relied on Col. 35, lines 24-65 from Kadambi to make a prior art showing of appellant's claimed technique "wherein said first data processing unit comprises logic for matching a first incoming packet to a stored first rule and for generating a first thread if the first incoming packet matches said first rule, said first thread identifying the location of one of said at least one data processing policies in said policy action table."

Appellant respectfully asserts that the excerpt from Kadambi relied upon by the Examiner merely teaches that "a logical AND operation is performed with the filter mask, having the selected fields enabled, and the packet" such that "[i]f there is a match, the matching entries are applied to rules tables 22, in order to determine which specific actions will be taken" (Col. 35, lines 24-28 – emphasis added). Further, the excerpt from Kadambi teaches that "since particular rules must be applied for various types of packets, the rules table requirements are minimized by setting all incoming packets to be 'tagged' packets" (Col. 35, lines 29-32 – emphasis added). Additionally, the excerpt from Kadambi teaches that "logic 1411 in FFP 141 determines and takes the appropriate action," and that "[t]he filtering logic can discard the packet, send the

packet to the CPU 52, modify the packet header or IP header, and recalculate any IP checksum fields or takes other appropriate action with respect to the headers" (Col. 35, lines 59-64 – emphasis added).

However, merely teaching that if there is a match, then the matching entries are applied to the rules table in order to determine which specific actions will be taken, in addition to suggesting that filtering logic can determine and take the appropriate action such as discarding the packet, sending the packet to the CPU, modifying the packet header, and recalculating any IP checksum fields, as in Kadambi, simply fails to suggest a technique "wherein said first data processing unit comprises logic for matching a first incoming packet to a stored first rule and for generating a first thread if the first incoming packet matches said first rule, said first thread identifying the location of one of said at least one data processing policies in said policy action table" (emphasis added), as claimed by appellant.

Clearly, applying matching entries to the rules table to determine which specific actions will be taken, in addition to filtering logic determining and taking the appropriate actions, as in Kadambi, simply fails to even suggest "generating a first thread," much less "generating a first thread if the first incoming packet matches said first rule, said first thread identifying the location of one of said at least one data processing policies in said policy action table" (emphasis added), as claimed by appellant. Furthermore, suggesting that the rules table requirements are minimized by setting all incoming packets to be 'tagged' packets, as in Kadambi, also simply fails to suggest "generating a first thread if the first incoming packet matches said first rule, said first thread identifying the location of one of said at least one data processing policies in said policy action table" (emphasis added), as claimed by appellant.

In the Examiner's Answer mailed 04/29/2008, the Examiner has stated that "[o]nce again, the [a]ppellant does not provide any arguments and only states that the art does not disclose the claim" and that "[t]here is no explanation as to why the [a]ppellant disagrees with the Examiner's rejection." Further, the Examiner has argued that "[i]n column 35, lines 24-65, packets are tagged for processing if they match selected fields in a filter mask" and that "[t]he matched packets are applied to rules tables to determine which actions will be taken." Additionally, the Examiner has argued that "[t]he thread generation and assignment is shown in

Irwin and the policy action table is shown in column 35 of Kadambi, both previous responses in this answer." In addition, the Examiner has stated that "[t]hus, the combination of Irwin and Kadambi discloses this group of claims."

Appellant respectfully disagrees and asserts that the arguments hereinabove clearly indicate as to how the excerpt from Kadambi relied upon by the Examiner fails to meet appellant's claimed technique. Further, appellant again asserts that the excerpt from Kadambi relied upon by the Examiner merely teaches that "a logical AND operation is performed with the filter mask, having the selected fields enabled, and the packet" such that "[i]f there is a match, the matching entries are applied to rules tables 22, in order to determine which specific actions will be taken" (Col. 35, lines 24-28 – emphasis added). Further, the except teaches that "[t]he filtering logic can discard the packet, send the packet to the CPU 52, modify the packet header or IP header, and recalculate any IP checksum fields or tak[e] other appropriate action with respect to the headers" (Col. 35, lines 60-64 – emphasis added).

However, teaching that if there is a match, the matching entries are applied to rules tables in order to determine which specific actions will be taken, where the filtering logic can discard the packet, send the packet to the CPU, modify the packet header or IP header, and recalculate any IP checksum fields, or take other appropriate action with respect to the headers, as in Kadambi, simply fails to suggest "generating a first thread **if** the first incoming packet matches said first rule," much less that "said first thread identifying the location of one of said at least one data processing policies in said policy action table" (emphasis added), as claimed by appellant. Clearly, applying matching entries to rules tables in order to determine which specific actions will be taken if there is a match, as in Kadambi, simply fails to suggest "generating a first thread **if** the first incoming packet matches said first rule, said first thread identifying the location of one of said at least one data processing policies in said policy action table" (emphasis added), as claimed by appellant.

Furthermore, appellant asserts that Irwin teaches that "[a]s each data packet enters the data packet processing system, a program counter is assigned to the packet for the duration of the time the packet is held in the data packet processing system" and that "[s]tepping through the program counter temporarily creates a virtual CPU execution unit, or a hardware thread, for the

data packet" (Col. 6, lines 5-13 – emphasis added). However, assigning a program counter to each data packet that enters the data packet processing system, and stepping through the program counter to temporarily create a hardware thread for the data packet, as in Irwin, simply fails to suggest "generating a first thread **if** the first incoming packet matches said first rule," much less that "said first thread identifying the location of one of said at least one data processing policies in said policy action table" (emphasis added), as claimed by appellant. Clearly, assigning a program counter **to each** data packet and stepping through the program counter to temporarily create a hardware thread for the data packet, as in Irwin, simply fails to even suggest "generating a first thread **if** the first incoming packet matches said first rule" (emphasis added), as claimed by appellant.

Again, appellant respectfully asserts that at least the third element of the *prima facie* case of obviousness has not been met, since the prior art excerpts, as relied upon by the Examiner, fail to teach or suggest all of the claim limitations, as noted above.

*Group #5: Claim 15*

With respect to Claim 15, the Examiner has relied on Col. 31, lines 34-45 and Col. 35, lines 24-65 of Kadambi to make a prior art showing of appellant's claimed "memory unit connected to said first data processing unit and to said second data processing unit, said memory unit adapted to temporarily store packets before processing by said second data processing unit." Specifically, the Examiner has argued that "Kadambi discloses packets stored within FFP."

Appellant respectfully disagrees and asserts that the excerpts from Kadambi relied upon by the Examiner teach that "FFP 141 includes filtering logic 1411, ...which selectively parses predetermined fields from the incoming data packets, thereby effectively obtaining the values of the desired fields from the MAC, IP, TCP, and UDP headers" (Col. 31, lines 41-45 – emphasis added). Further, the excerpts teach that "FFP 141 is shown to include filter database 1410 containing filter masks therein" (Col. 35, lines 46-47 – emphasis added).

Therefore, in Kadambi, the FFP is taught to include filtering logic that selectively parses predetermined fields from incoming data packets, and a filter database containing filter masks,

which simply fails to meet appellant's claimed "memory unit connected to said first data processing unit and to said second data processing unit, said memory unit adapted to temporarily store packets before processing by said second data processing unit" (emphasis added), as claimed by appellant. Clearly, the FFP including filtering logic and a filter database, as in Kadambi, simply fails to meet "said memory unit adapted to temporarily store packets before processing by said second data processing unit" (emphasis added), as claimed by appellant.

In the Examiner's Answer mailed 04/29/2008, the Examiner has argued that "[i]n column 30, lines 54-66 of Kadambi there is the use of priority queues that temporarily store packets" and that "column 10, lines 5-17, of Irwin discloses a data packet stored in a packet buffer before processing by the second data processing unit." Further, the Examiner has stated that "Claim 15 is clearly shown in the combination of arts."

Appellant respectfully disagrees and asserts that the excerpt from Kadambi relied upon by the Examiner merely teaches that "[f]ilters are used for packet classification" and "[v]arious actions are taken based upon the packet classification, including packet discard, sending of the packet to the CPU, sending of the packet to other ports, sending the packet on certain COS [(class of service)] priority queues, and changing the type of service (TOS) precedence" (Col. 30, lines 59-66 – emphasis added). Further, the excerpt, in addition to Fig. 21, from Irwin teaches that "[a]s each data packet is received, the master node stores the data packet in a packet buffer (S02), and assigns a program counter to the data packet (S04)" (Col. 10, lines 5-7 – emphasis added).

However, teaching that the data packet is stored in a packet buffer as each packet is received, as in Irwin, in addition to teaching that based upon the packet classification, various actions are taken including sending the packet on certain class of service priority queues, as in Kadambi, simply fails to meet appellant's claimed "memory unit connected to said first data processing unit and to said second data processing unit, said memory unit adapted to temporarily store packets **before processing** by said second data processing unit" (emphasis added), as claimed by appellant. Clearly, storing a data packet in a packet buffer as it is received, as in Irwin, in addition to sending the packet on certain class of service priority queues based upon the packet classification, as in Kadambi, simply fails to even suggest a "memory unit connected to said first data processing unit and to said second data processing unit" where "said memory unit [is]

adapted to temporarily store packets **before processing** by said second data processing unit" (emphasis added), in the context claimed by appellant.

Again, appellant respectfully asserts that at least the third element of the *prima facie* case of obviousness has not been met, since the prior art excerpts, as relied upon by the Examiner, fail to teach or suggest all of the claim limitations, as noted above.

*Group #6: Claim 30*

With respect to Claim 30, the Examiner has relied on the following excerpts from Irwin to meet appellant's claimed "first data processing unit adapted to filter incoming packets."

> "The data packet processing is performed by executing a packet switching software of the router. The processing for forwarding a data packet can be broken into a number of procedures. Typically, packet forwarding requires procedures, such as translation of a header of the data packet to identify suitable routes, classification of a packet flow according to source and destination addresses..." (Col. 5, lines 46-52 - emphasis added)
>
> "[serv]ice route pruning, metric route pruning, policy route pruning, option processing, congestion control, scheduling, and performance monitoring." (Col. 5, lines 53-55 - not specifically cited - emphasis added)
>
> "As each data packet enters the data packet processing system, a program counter is assigned to the packet for the duration of the time the packet is held in the data packet processing system." (Col. 6, lines 5-8)

Appellant respectfully asserts that the excerpts from Irwin relied upon by the Examiner merely disclose that "[t]he processing for forwarding a data packet can be broken into a number of procedures such as translation of a header of the data packet to identify suitable routes, classification of a packet flow according to source and destination addresses, type of service route pruning, metric route pruning, policy route pruning, option processing, congestion control, scheduling, and performance monitoring" (emphasis added). However, merely disclosing that forwarding a data packet may be broken into procedures such as the translation of a header to identify suitable routes, and classification of a packet flow according to source and destination addresses, as in Irwin, fails to suggest "a first data processing unit adapted to filter incoming packets" (emphasis added), as claimed by appellant. Clearly, procedures for forwarding a data

packet, as in Irwin, simply fails to even suggest "filter[ing] incoming packets," as claimed by appellant.

In the Examiner's Answer mailed 04/29/2008, the Examiner has argued that "[a]s seen in column 5, lines 47-65, the forwarding program determines the outgoing route to be followed by a data packet" and that "[t]he definition of filtering is a device or program that separates data, which is exactly what the forwarding of Irwin is doing." Further, the Examiner has argued that "[t]he packets are forwarded to different routes, thus being separated, which is the very definition of filter." In addition, the Examiner has argued that "[t]he data packet processing system, on column 6, lines 5-8, assigns a program counter to incoming packets and forwards them using the forwarding program; thus Irwin clearly teaches 'a first data processing unit adapted to filter incoming packets.'"

Appellant respectfully disagrees and asserts that the excerpts from Irwin relied upon by the Examiner merely teach "[t]he processing for forwarding a data packet can be broken into a number of procedures" and that "[t]ypically, packet forwarding requires procedures, such as translation of a header of the data packet to identify suitable routes, classification of a packet flow according to source and destination addresses, type of service route pruning, metric route pruning, policy route pruning, option processing, congestion control, scheduling, and performance monitoring" (Col. 5, lines 47-55 – emphasis added). Further, the excerpts teach that "a program counter is assigned to the packet for the duration of the time the packet is held in the data packet processing system" (Col. 6, lines 5-8 – emphasis added).

However, the disclosure that packet forwarding requires procedures such as translation of a header to identify suitable routes, classification of a packet, route pruning, option processing, congestion control, scheduling, and performance monitoring, in addition to the disclosure of assigning a program counter to the packet, as in Irwin, fails to suggest "a first data processing unit adapted to filter incoming packets" (emphasis added), as claimed by appellant. Clearly, procedures for packet forwarding, as in Irwin, simply fail to even suggest "filter[ing] incoming packets," as claimed by appellant.

Additionally, appellant respectfully asserts that it would be unobvious to modify Irwin to "filter incoming packets" (emphasis added), as claimed by appellant. Specifically, Irwin teaches "a router having forwarding engines at network interfaces" where "[e]ach interface 12 may be provisioned with a forwarding engine 18 that processes incoming packets in order to determine which outgoing network interface 12 needs to be accessed through the interconnect structure of the switching fabric 14" (Col. 1, lines 32-40 – emphasis added). Further, Irwin teaches that "[s]ince the network processor 16 is no longer required to make routing decisions for a data packet, the packet throughput is capable of being scaled with the number of physical interfaces 12" such that "[t]he single point of congestion as formerly experienced with the centralization of the routing algorithms is replaced by multiple processors 12" (Col. 1, lines 52-57 – emphasis added).

Appellant asserts that Irwin's router with forwarding engines at each network interface simply utilizes the forwarding engines to determine which outgoing network interface needs to be accessed through the interconnect structure in order to alleviate the network processor from making routing decisions, which allows the throughput to be scaled with each interface. Clearly, a router with a forwarding engine at each interface that determines which outgoing network interface needs to be accessed through the interconnect structure, would be unable to filter incoming packets due to the congestion as formerly experienced with the centralization of the routing algorithms, as expressly noted in Irwin. Therefore, appellant asserts that it would be unobvious to include filtering, in the manner claimed by appellant, in Irwin's multiple network interface router.

Further, in the Examiner's Answer mailed 04/29/2008, the Examiner has argued that "[a]ppellant also relies on column 1, lines 32-57 of Irwin to show that the limitation is not found, but the Examiner would like to point out, regardless, that portion of Irwin refers to figure 1 which is labeled as prior art" and "[e]ven so, it teaches forwarding through different interfaces which is clearly filtering."

Appellant respectfully disagrees and asserts that Fig. 1 of Irwin clearly "shows an example of a router having forwarding engines at network interfaces" and that "[t]he router 10 consists of a plurality of network interfaces" where "[e]ach interface 12 may be provisioned with a

forwarding engine 18 that processes incoming packets in order to determine which outgoing network interface 12 needs to be accessed through the interconnect structure of the switching fabric 14" (Col. 1, lines 32-40 – emphasis added).

Again, appellant asserts that a router with a forwarding engine at each interface that determines which outgoing network interface needs to be accessed through the interconnect structure, as in Irwin, simply fails to suggest "a first data processing unit adapted to filter incoming packets" (emphasis added), as claimed by appellant. Clearly, a forwarding engine determining which outgoing network interface needs to be accessed, as in Irwin, simply fails to even suggest "filter[ing] incoming packets," as claimed by appellant.

Further, with respect to Claim 30, the Examiner has relied on the following excerpt from Irwin to meet appellant's claimed "a second data processing unit adapted to process incoming packets according to one of said plurality of instruction sets after the filtering, based on a thread assigned to the incoming packets by the first data processing unit."

> "As each data packet enters the data packet processing system, a program counter is assigned to the packet for the duration of the time the packet is held in the data packet processing system. The program counter uniquely defines a main program flow in the main forwarding program that is executed for the packet to select an output queue of the router for routing the packet. Stepping through the program counter temporarily creates a virtual CPU execution unit, or a hardware thread, for the data packet. Each thread is represented by a thread identification number which is defined by the program counter and a register file used by the process followed for packet processing." (Col. 6, lines 5-16 - emphasis added)

Appellant respectfully asserts that the excerpt from Irwin relied upon by the Examiner merely teaches that "[a]s each data packet enters the data packet processing system, a program counter is assigned to the packet for the duration of the time the packet is held in the data packet processing system" and that "[s]tepping through the program counter temporarily creates a virtual CPU execution unit, or a hardware thread, for the data packet" (emphasis added).

However, the mere disclosure of assigning a program counter to the packet, such that stepping through the program counter creates a virtual CPU execution unit or hardware thread for the data packet, as in Irwin, fails to teach "a second data processing unit adapted to process incoming

packets according to one of said plurality of instruction sets after the filtering, based on a thread assigned to the incoming packets by the first data processing unit" (emphasis added), as claimed by appellant. Clearly, stepping through the program counter to create a hardware thread for the packet, as in Irwin, simply fails to even suggest "process[ing] incoming packets according to one of said plurality of instruction sets **after the filtering**, based on a thread" (emphasis added), as claimed by appellant.

Additionally, appellant respectfully asserts that Irwin, as argued above, simply fails to suggest any sort of "filtering," as claimed by appellant, let alone in the specific context claimed by appellant. For example, appellant asserts that Irwin merely discloses that "[w]hen the end procedural call is read and the end of the main forwarding program has been reached (S32), if there is no data packet to be processed (S34), the thread is placed in the queue of idle threads since the packet processing has been completed and the packet has been forwarded to the designated route" (Col. 10, lines 56-61 – emphasis added). Further, Irwin teaches that "[i]f there is a data packet to be processed (S34), the thread is reactivated for a new packet (S02)" (Col. 10, lines 61-62).

However, disclosing that the thread is placed in the queue of idle threads after the packet processing has been completed and the packet has been forwarded to the designated route, as in Irwin, fails to suggest "a second data processing unit adapted to process incoming packets according to one of said plurality of instruction sets after the filtering, based on a thread assigned to the incoming packets by the first data processing unit" (emphasis added), as claimed. Clearly, placing the thread in the queue of idle threads after processing is completed, as in Irwin, simply fails to even suggest "process[ing] incoming packets according to one of said plurality of instruction sets **after the filtering**, **based on a thread assigned** to the incoming packets by the first data processing unit" (emphasis added), as claimed by appellant.

In the Examiner's Answer mailed 04/29/2008, the Examiner has argued that "[a]s seen in column 6, lines 5-16, a thread is assigned to the packet by the data packet processing system, which is the first data processing unit of the claim." Further, the Examiner has argued that "[t]he second data processing unit is in the process followed after the packet process" and that "[t]his second data processing unit uses the thread which is represented by a thread identification number." In

addition, the Examiner has argued that "[t]hus, column 6, lines 5-16, clearly teaches this limitation."

Appellant respectfully disagrees and asserts that Irwin teaches that "[a] data packet processing system for a router in accordance with the present invention uses a multiprocessor architecture where a sequence of operations is required to be executed to determine a route which an incoming data packet needs to follow in order to reach its destination" (Col. 5, lines 34-38, not specifically cited – emphasis added).

However, a data packet processing system for a router that determines a route which an incoming data packet needs to follow in order to reach its destination, as in Irwin, simply fails to even suggest "process[ing] incoming packets according to one of said plurality of instruction sets **after the filtering**" (emphasis added), as claimed by appellant. Clearly, the data packet processing system that determines a route for an incoming data packet, as in Irwin, simply fails to even suggest "process[ing] incoming packets according to one of said plurality of instruction sets **after the filtering**" (emphasis added), as claimed by appellant.

Further, the excerpt from Irwin relied upon by the Examiner teaches that "[a]s each data packet enters the data packet processing system, a program counter is assigned to the packet for the duration of the time the packet is held in the data packet processing system" and that "[s]tepping through the program counter temporarily creates a virtual CPU execution unit, or a hardware thread, for the data packet," where "[e]ach thread is represented by a thread identification number which is defined by the program counter and a register file used by the process followed for packet processing" (Col. 6, lines 5-16 – emphasis added). Additionally, Irwin teaches that in "[s]tepping through the program counter, an output queue of the router is selected for routing the packet" and that "[w]hen packet processing has been completed, the packet has been forwarded to the selected output queue and the thread is made idle or destroyed" (Col. 6, lines 29-33 – emphasis added).

However, assigning a program counter to the packet, stepping through the program counter to temporarily create a hardware thread for the data packet, where each thread is represented by a thread identification number defined by the program counter and a register file used by the

process for packet processing, stepping through the program counter to select an output queue for routing the packet, and idling or destroying the thread **after the packet has been forwarded** to the selected output queue, as in Irwin, simply fails to suggest "process[ing] incoming packets according to one of said plurality of instruction sets **after the filtering**, **based on a thread assigned** to the incoming packets by the first data processing unit" (emphasis added), as claimed by appellant. Clearly, stepping through the program counter to temporarily create a hardware thread for the data packet, stepping through the program counter to select an output queue for routing the packet, and idling or destroying the thread **after the packet has been forwarded**, as in Irwin, simply fails to even suggest "process[ing] incoming packets according to one of said plurality of instruction sets **after the filtering**, **based on a thread assigned** to the incoming packets" (emphasis added), as claimed by appellant.

In addition, with respect to Claim 30, the Examiner has relied on the following excerpts from Irwin to meet appellant's claimed "a data bus connecting the addressable memory unit and the first and second data processing units."

> "The received requests are forwarded to a processing unit (not shown) via a processor bus 119 using a node dependent logic 132 for processing the requests." (Col. 8, lines 49-52 - emphasis added)
>
> "FIG. 8 illustrates the P1394.2 functional blocks of a transmission interface that may be used in a computing node for a multiprocessor array used in the present invention. The computing array has an east to west macro 118 and a west to east macro 120 which are connected by a processor bus 119 to a processing unit (not shown). The terms "east" and "west" are used for convenience to indicate two opposite directions in a ring topology." (Col. 8, lines 29-36 - not specifically cited - emphasis added)

Appellant respectfully asserts that the excerpt from Irwin relied upon by the Examiner simply teaches that "received requests are forwarded to a processing unit (not shown) via a processor bus 119" (emphasis added). Further, Irwin teaches that "[t]he computing array has an east to west macro 118 and a west to east macro 120 which are connected by a processor bus 119 to a processing unit (not shown)" (Col. 8, lines 31-34 – emphasis added). However, the mere disclosure of a processing unit connected by a processor bus to an east to west macro, and a west to east macro, as in Irwin, fails to teach "a data bus connecting the addressable memory unit and the first and second data processing units" (emphasis added), as claimed by appellant.

In the Examiner's Answer mailed 04/29/2008, the Examiner has argued that "column 8, lines 49-52... show that a data bus is explicitly shown in the reference" and that "[t]here is also a processor bus shown on column 7, lines 49-52." Further, the Examiner has argued that "it is clear from reading column 6, lines 5-16, that the addressable memory unit and first and second processing units are connected by a data bus as they all load data and data is transferred from one to another and thus must be connected by a data bus" and that "[t]he data bus is inherently necessary."

Appellant respectfully disagrees and asserts that the excerpts from Irwin relied upon by the Examiner merely teach that "[t]he received requests are forwarded to a processing unit (not shown) via a processor bus 119 using a node dependent logic 132 for processing the requests" (Col. 8, lines 49-52 – emphasis added). Further, the excerpts from Irwin teach "[a] packet switching software of the router which is executed by the data packet processing system 100" (Col. 7, lines 49-50), where "[t]he data packet processing system 100 comprises two I/O nodes 102 as master nodes, and a processor array having three computing nodes 110 as slave nodes which are connected to the I/O nodes 102 in a ring topology" (Col. 7, lines 42-45 – emphasis added). In addition, the excerpts teach that "a program counter is assigned to the packet for the duration of the time the packet is held in the data packet processing system" (Col. 6, lines 5-8 – emphasis added).

However, teaching that received requests are forwarded to a processing unit via a processor bus, and that a data packet processing system comprises two I/O nodes 102 and a processor array having three computing nodes 110 connected to the I/O nodes 102 in a ring topology, in addition to teaching that a program counter is assigned to the packet, as in Irwin, simply fails to even suggest "a data bus connecting the addressable memory unit and the first and second data processing units" (emphasis added), as claimed by appellant.

Furthermore, Fig. 7 of Irwin shows that each computing node 110 of the processing array includes Memory, CPU, and Cache. Additionally, Fig. 7 of Irwin shows that each I/O node 102 includes I/O, Memory, and CCU. Clearly, computing nodes 110 that each include memory, in addition to I/O nodes 102 that each include memory, as in Irwin, simply fails to support the Examiner's allegation that "the addressable memory unit and first and second processing units

are connected by a data bus as they all load data and data is transferred from one to another and thus must be connected by a data bus" (emphasis added), as alleged by the Examiner. In addition, showing that computing nodes 110 and I/O nodes 102 each include memory, as in Irwin, simply fails to even suggest "a data bus connecting the addressable memory unit and the first and second data processing units" (emphasis added), as claimed by appellant.

Additionally, it appears that the Examiner has relied on an inherency argument regarding the above emphasized claim limitations. In view of the arguments made hereinabove, any such inherency argument has been adequately rebutted, and a notice of allowance or a specific prior art showing of such claim features, in combination with the remaining claim elements is respectfully requested. (See MPEP 2112)

Further, in response, appellant asserts that the fact that a certain result or characteristic may occur or be present in the prior art is not sufficient to establish the inherency of that result or characteristic. *In re Rijckaert*, 9 F.3d 1531, 1534, 28 USPQ2d 1955, 1957 (Fed. Cir. 1993); *In re Oelrich*, 666 F.2d 578, 581-82, 212 USPQ 323, 326 (CCPA 1981). "To establish inherency, the extrinsic evidence 'must make clear that the missing descriptive matter is necessarily present in the thing described in the reference, and that it would be so recognized by persons of ordinary skill. Inherency, however, may not be established by probabilities or possibilities. The mere fact that a certain thing may result from a given set of circumstances is not sufficient.'" *In re Robertson*, 169 F.3d 743, 745, 49 USPQ2d 1949, 1950-51 (Fed. Cir. 1999).

Additionally, with respect to Claim 30, the Examiner has relied on the following excerpt to make a prior art showing of appellant's claimed technique "wherein a policy action table is connected to said data bus and said addressable memory unit, wherein said policy action table stores at least one data processing policy."

> "Referring once again to FIG. 14, after. FFP 141 applies appropriate configured filters and results are obtained from the appropriate rules table 22, logic 1411 in FFP 141 determines and takes the appropriate action. The filtering logic can discard the packet, send the packet to the CPU 52, modify the packet header or IP header, and recalculate any IP checksum fields or takes other appropriate action with respect to the headers. The modification occurs at buffer slicer 144, and the packet is placed on C channel 81." (Kadambi - Col. 35, lines 57-64 - emphasis added)

Appellant respectfully asserts that the excerpt form Kadambi relied upon by the Examiner simply teaches that "[t]he filtering logic can discard the packet, send the packet to the CPU 52, modify the packet header or IP header, and recalculate any IP checksum fields or takes other appropriate action with respect to the headers" (emphasis added). However, the mere disclosure of filtering logic that can discard the packet, send the packet to the CPU, modify the packet header, and recalculate any IP checksum fields, as in Kadambi, fails to teach a "policy action table," much less that "a policy action table is connected to said data bus and said addressable memory unit, wherein said policy action table stores at least one data processing policy" (emphasis added), as claimed by appellant. Clearly, filtering logic, as in Kadambi, simply fails to even suggest a "policy action table [that] stores at least one data processing policy" (emphasis added), as claimed by appellant.

In the Examiner's Answer mailed 04/29/2008, the Examiner has argued that "column 35, lines 57-64, teaches a rules table that stores the filtering logic used to take appropriate action with" and that "[t]he rules table is equivalent to a policy action table as the filtering logic is a data processing policy." Additionally, the Examiner has argued that "[t]he filtering logic is a processing policy because it has policies for packet processing as evidenced by the various processing actions the filtering logic can perform." Furthermore, the Examiner has concluded that "[t]hus, [appellant's claimed technique] is clearly disclosed in combination of Irwin and Kadambi."

Appellant respectfully disagrees and asserts that the excerpt from Irwin relied upon by the Examiner merely teaches that "[e]ach interface 12 may be provisioned with a forwarding engine 18 that processes incoming packets in order to determine which outgoing network interface 12 needs to be accessed through the interconnect structure of the switching fabric 14" and that "a local version of the forwarding algorithms and data tables found normally in the network processor are downloaded from the network processor 16 to the forwarding engine 18 of the network interface 12" (Col. 1, lines 33-48 – emphasis added). Further, Irwin teaches that "[f]orwarding of the IP datagram requires the router to lookup the destination address in a forwarding table to identify which interface the IP datagram is to be forwarded" (Col. 1, lines 19-22 – emphasis added).

However, merely teaching that forwarding algorithms and data tables are downloaded from the network processor to the forwarding engine, and the router looks up the destination address in a forwarding table to identify an interface, as in Irwin, simply fails to suggest "a policy action table," much less that "a policy action table is connected to said data bus and said addressable memory unit, wherein said policy action table stores at least one data processing policy" (emphasis added), as claimed by appellant. Clearly, looking up a destination address in a forwarding table to identify an interface, as in Irwin, simply fails to even suggest that "said policy action table stores at least one data processing policy" (emphasis added), as claimed by appellant.

In addition, appellant asserts that the excerpt from Kadambi relied upon by the Examiner teaches that "after... FFP [(fast filtering processor)] 141 applies appropriate configured filters and results are obtained from the appropriate rules table 22, logic 1411 in FFP 141 determines and takes the appropriate action" and that "[t]he filtering logic can discard the packet, send the packet to the CPU 52, modify the packet header or IP header, and recalculate any IP checksum fields or takes other appropriate action with respect to the headers" (Col. 35, lines 57-64 – emphasis added).

However, obtaining results from the rules table after the fast filtering processor applies configured filters, in addition to the filtering logic discarding the packet, sending the packet to the CPU, modifying the packet header or IP header, and recalculating any IP checksum fields or taking other appropriate action with respect to the headers, as in Kadambi, simply fails to suggest "a policy action table," much less that "a policy action table is connected to said data bus and said addressable memory unit, wherein said policy action table stores at least one data processing policy" (emphasis added), as claimed by appellant. Clearly, obtaining results from the rules table after the fast filtering processor applies configured filters, as in Irwin, simply fails to even suggest that "said policy action table stores at least one data processing policy" (emphasis added), as claimed by appellant.

Furthermore, with respect to Claim 30, the Examiner has relied on Col. 35, lines 24-65 from Kadambi to make a prior art showing of appellant's claimed technique "wherein said first data processing unit comprises logic for matching a first incoming packet to a stored first rule and for

generating a first thread if the first incoming packet matches said first rule, said first thread identifying the location of one of said at least one data processing policies in said policy action table."

Appellant respectfully asserts that the excerpt from Kadambi relied upon by the Examiner merely teaches that "a logical AND operation is performed with the filter mask, having the selected fields enabled, and the packet" such that "[i]f there is a match, the matching entries are applied to rules tables 22, in order to determine which specific actions will be taken" (Col. 35, lines 24-28 – emphasis added). Further, the excerpt from Kadambi teaches that "since particular rules must be applied for various types of packets, the rules table requirements are minimized by setting all incoming packets to be 'tagged' packets" (Col. 35, lines 29-32 – emphasis added). Additionally, the excerpt from Kadambi teaches that "logic 1411 in FFP 141 determines and takes the appropriate action," and that "[t]he filtering logic can discard the packet, send the packet to the CPU 52, modify the packet header or IP header, and recalculate any IP checksum fields or takes other appropriate action with respect to the headers" (Col. 35, lines 59-64 – emphasis added).

However, merely teaching that if there is a match, then the matching entries are applied to the rules table in order to determine which specific actions will be taken, in addition to suggesting that filtering logic can determine and take the appropriate action such as discarding the packet, sending the packet to the CPU, modifying the packet header, and recalculating any IP checksum fields, as in Kadambi, simply fails to suggest a technique "wherein said first data processing unit comprises logic for matching a first incoming packet to a stored first rule and for generating a first thread if the first incoming packet matches said first rule, said first thread identifying the location of one of said at least one data processing policies in said policy action table" (emphasis added), as claimed by appellant.

Clearly, applying matching entries to the rules table to determine which specific actions will be taken, in addition to filtering logic determining and taking the appropriate actions, as in Kadambi, simply fails to even suggest "generating a first thread," much less "generating a first thread if the first incoming packet matches said first rule, said first thread identifying the location of one of said at least one data processing policies in said policy action table" (emphasis added),

as claimed by appellant. Furthermore, suggesting that the rules table requirements are minimized by setting all incoming packets to be 'tagged' packets, as in Kadambi, also simply fails to suggest "generating a first thread if the first incoming packet matches said first rule, said first thread identifying the location of one of said at least one data processing policies in said policy action table" (emphasis added), as claimed by appellant.

In the Examiner's Answer mailed 04/29/2008, the Examiner has argued that "[at] column 35, lines 24-65, packets are tagged for processing if they match selected fields in a filter mask" and that "[t]he matched packets are applied to rules tables to determine which actions will be taken." Additionally, the Examiner has argued that "[t]he thread generation and assignment is shown in Irwin and the policy action table is shown in column 35 of Kadambi, both previous responses in this answer." In addition, the Examiner has stated that "[t]hus, the combination of Irwin and Kadambi discloses this limitation."

Appellant asserts that the excerpt from Kadambi relied upon by the Examiner merely teaches that "a logical AND operation is performed with the filter mask, having the selected fields enabled, and the packet" such that "[i]f there is a match, the matching entries are applied to rules tables 22, in order to determine which specific actions will be taken" (Col. 35, lines 24-28 – emphasis added). Further, the except teaches that "[t]he filtering logic can discard the packet, send the packet to the CPU 52, modify the packet header or IP header, and recalculate any IP checksum fields or takes other appropriate action with respect to the headers" (Col. 35, lines 60-64 – emphasis added).

However, teaching that if there is a match, the matching entries are applied to rules tables in order to determine which specific actions will be taken, where the filtering logic can discard the packet, send the packet to the CPU, modify the packet header or IP header, and recalculate any IP checksum fields, or take other appropriate action with respect to the headers, as in Kadambi, simply fails to suggest "generating a first thread if the first incoming packet matches said first rule," much less "said first thread identifying the location of one of said at least one data processing policies in said policy action table" (emphasis added), as claimed by appellant. Clearly, applying matching entries to rules tables in order to determine which specific actions will be taken if there is a match, as in Kadambi, simply fails to suggest "generating a first thread

**if** the first incoming packet matches said first rule, said first thread identifying the location of one of said at least one data processing policies in said policy action table" (emphasis added), as claimed by appellant.

Furthermore, appellant asserts that Irwin teaches that "[a]s each data packet enters the data packet processing system, a program counter is assigned to the packet for the duration of the time the packet is held in the data packet processing system" and that "[s]tepping through the program counter temporarily creates a virtual CPU execution unit, or a hardware thread, for the data packet" (Col. 6, lines 5-13 – emphasis added). However, assigning a program counter to each data packet that enters the data packet processing system, and stepping through the program counter to temporarily create a hardware thread for the data packet, as in Irwin, simply fails to suggest "generating a first thread **if** the first incoming packet matches said first rule," much less "said first thread identifying the location of one of said at least one data processing policies in said policy action table" (emphasis added), as claimed by appellant. Clearly, assigning a program counter to **each** data packet and stepping through the program counter to temporarily create a hardware thread for the data packet, as in Irwin, simply fails to even suggest "generating a first thread **if** the first incoming packet matches said first rule" (emphasis added), as claimed by appellant.

Still yet, with respect to Claim 30, the Examiner has relied on Col. 31, lines 20-34 and Col. 35, lines 57-64 from Kadambi to make a prior art showing of appellant's claimed technique "wherein said data processing policy comprises a first address pointer to a starting address of a first set of instructions and a second address pointer to a starting address of a state block stored in said addressable memory unit, said state block used by said first set of instructions for processing the first incoming packet."

Appellant respectfully asserts that the Col. 35, lines 57-64 from Kadambi relied upon by the Examiner teaches that "logic 1411 in FFP 141 determines and takes the appropriate action," and that "[t]he filtering logic can discard the packet, send the packet to the CPU 52, modify the packet header or IP header, and recalculate any IP checksum fields or takes other appropriate action with respect to the headers" (emphasis added). However, the mere disclosure of filtering logic that can determine and take the appropriate action such as discarding the packet, sending

the packet to the CPU, modifying the packet header, and recalculating any IP checksum fields, as in Kadambi, fails to suggest a "first address pointer...and a second address pointer," much less a technique "wherein said data processing policy comprises a first address pointer to a starting address of a first set of instructions and a second address pointer to a starting address of a state block stored in said addressable memory unit, said state block used by said first set of instructions for processing the first incoming packet" (emphasis added), as claimed by appellant.

In addition, appellant respectfully asserts that Col. 31, lines 20-34 from Kadambi relied upon by the Examiner simply teaches that "FFP 141 is essentially a state machine driven programmable rules engine" (emphasis added). However, the mere disclosure of a state machine driven programmable rules engine, as in Kadambi, simply fails to suggest any sort of a "first address pointer...and a second address pointer," let alone a technique "wherein said data processing policy comprises a first address pointer to a starting address of a first set of instructions and a second address pointer to a starting address of a state block stored in said addressable memory unit, said state block used by said first set of instructions for processing the first incoming packet" (emphasis added), as claimed by appellant.

In the Examiner's Answer mailed 04/29/2008, the Examiner has argued that "[at] column 35, lines 57-64, Kadambi discloses logic 1411 in the FFP 141 which points to instruction sets to take action and, [at] column 31, lines 20-34, Kadambi discloses the FFP which is essentially a state machine." In addition, the Examiner has argued that "Irwin discloses the addressable memory unit and processing a first incoming packet and thus, it is shown that Kadambi combined with Irwin discloses this limitation."

Appellant again asserts that the excerpts from Kadambi relied upon by the Examiner teach that "logic 1411 in FFP 141 determines and takes the appropriate action" (Col. 35, lines 57-60 – emphasis added). Further, the excerpts teach that that "FFP 141 is essentially a state machine driven programmable rules engine" (Col. 31, lines 20-21 –emphasis added). Clearly, logic 1411 determining and taking the appropriate action, in addition to a programmable rules engine driven by a state machine, as in Kadambi, simply fails to support the Examiner's allegation that "Kadambi discloses logic 1411 in the FFP 141 which points to instruction sets to take action" (emphasis added), as alleged by the Examiner. Furthermore, logic determining and taking the

appropriate action, in addition to a programmable rules engine driven by a state machine, as in Kadambi, fails to suggest a "first address pointer...and a second address pointer," let alone a technique "wherein said data processing policy comprises a first address pointer to a starting address of a first set of instructions and a second address pointer to a starting address of a state block stored in said addressable memory unit, said state block used by said first set of instructions for processing the first incoming packet" (emphasis added), as claimed by appellant.

Furthermore, appellant again asserts that Irwin merely teaches that "a program counter is assigned to the packet for the duration of the time the packet is held in the data packet processing system" (Col. 6, lines 5-8 – emphasis added). Clearly, assigning a program counter to the packet, as in Irwin, simply fails to suggest "a first address pointer to a starting address of a first set of instructions and a second address pointer to a starting address of a state block stored in said addressable memory unit" (emphasis added), as claimed by appellant. In addition, assigning a program counter to the packet, as in Irwin, simply fails to even suggest an "addressable memory unit," as claimed by appellant.

Additionally, with respect to Claim 30, the Examiner has relied on Col. 35, lines 24-65 from Kadambi to make a prior art showing of appellant's claimed technique "wherein said first processing unit further comprises logic for matching a second incoming packet to a stored second rule and for generating a second thread if the second incoming packet matches the second rule, said second thread identifying the location of one of said at least one data processing policy in said policy action table."

Appellant respectfully asserts that the excerpt from Kadambi relied upon by the Examiner merely teaches that "a logical AND operation is performed with the filter mask, having the selected fields enabled, and the packet" such that "[i]f there is a match, the matching entries are applied to rules tables 22, in order to determine which specific actions will be taken" (Col. 35, lines 24-28 – emphasis added). Further, the excerpt from Kadambi teaches that "since particular rules must be applied for various types of packets, the rules table requirements are minimized by setting all incoming packets to be 'tagged' packets" (Col. 35, lines 29-32 – emphasis added). Additionally, the excerpt from Kadambi teaches that "logic 1411 in FFP 141 determines and takes the appropriate action," and that "[t]he filtering logic can discard the packet, send the

packet to the CPU 52, modify the packet header or IP header, and recalculate any IP checksum fields or takes other appropriate action with respect to the headers" (Col. 35, lines 59-64 – emphasis added).

However, merely teaching that if there is a match, then the matching entries are applied to the rules table in order to determine which specific actions will be taken, in addition to suggesting that filtering logic can determine and take the appropriate action such as discarding the packet, sending the packet to the CPU, modifying the packet header, and recalculating any IP checksum fields, as in Kadambi, simply fails to suggest a technique "wherein said first processing unit further comprises logic for matching a second incoming packet to a stored second rule and for generating a second thread if the second incoming packet matches the second rule, said second thread identifying the location of one of said at least one data processing policy in said policy action table" (emphasis added), as claimed by appellant.

Clearly, applying matching entries to the rules table to determine which specific actions will be taken, in addition to filtering logic determining and taking the appropriate actions, as in Kadambi, simply fails to even suggest "generating a second thread," much less "generating a second thread if the second incoming packet matches the second rule, said second thread identifying the location of one of said at least one data processing policy in said policy action table" (emphasis added), as claimed by appellant. Furthermore, suggesting that the rules table requirements are minimized by setting all incoming packets to be 'tagged' packets, as in Kadambi, also simply fails to suggest "generating a second thread if the second incoming packet matches the second rule, said second thread identifying the location of one of said at least one data processing policy in said policy action table" (emphasis added), as claimed by appellant.

In the Examiner's Answer mailed 04/29/2008, the Examiner has argued that "[at] column 35, lines 24-65, packets are tagged for processing if they match selected fields in a filter mask" and that "[t]he matched packets are applied to rules tables to determine which actions will be taken." Additionally, the Examiner has argued that "[t]he multiple thread generation and assignment and multiple incoming packets is shown in Irwin and the policy action table is shown in column 35 of Kadambi, both previous responses in this answer." In addition, the Examiner has stated that "[t]hus, the combination of Irwin and Kadambi discloses this limitation."

Appellant again asserts that the excerpt from Kadambi relied upon by the Examiner merely teaches that "a logical AND operation is performed with the filter mask, having the selected fields enabled, and the packet" such that "[i]f there is a match, the matching entries are applied to rules tables 22, in order to determine which specific actions will be taken" (Col. 35, lines 24-28 – emphasis added). Further, the except teaches that "[t]he filtering logic can discard the packet, send the packet to the CPU 52, modify the packet header or IP header, and recalculate any IP checksum fields or takes other appropriate action with respect to the headers" (Col. 35, lines 60-64 – emphasis added).

However, teaching that if there is a match, the matching entries are applied to rules tables in order to determine which specific actions will be taken, where the filtering logic can discard the packet, send the packet to the CPU, modify the packet header or IP header, and recalculate any IP checksum fields, or take other appropriate action with respect to the headers, as in Kadambi, simply fails to suggest "generating a second thread **if** the second incoming packet matches the second rule," much less that "said second thread identifying the location of one of said at least one data processing policy in said policy action table" (emphasis added), as claimed by appellant. Clearly, applying matching entries to rules tables in order to determine which specific actions will be taken if there is a match, as in Kadambi, simply fails to suggest "generating a second thread **if** the second incoming packet matches the second rule, said second thread identifying the location of one of said at least one data processing policy in said policy action table" (emphasis added), as claimed by appellant.

Furthermore, appellant again asserts that Irwin teaches that "[a]s each data packet enters the data packet processing system, a program counter is assigned to the packet for the duration of the time the packet is held in the data packet processing system" and that "[s]tepping through the program counter temporarily creates a virtual CPU execution unit, or a hardware thread, for the data packet" (Col. 6, lines 5-13 – emphasis added). However, assigning a program counter to each data packet that enters the data packet processing system, and stepping through the program counter to temporarily create a hardware thread for the data packet, as in Irwin, simply fails to suggest "generating a second thread if the second incoming packet matches the second rule, said second thread identifying the location of one of said at least one data processing policy in said

policy action table" (emphasis added), as claimed by appellant. Clearly, assigning a program counter to each data packet and stepping through the program counter to temporarily create a hardware thread for the data packet, as in Irwin, simply fails to even suggest "generating a second thread **if** the second incoming packet matches the second rule" (emphasis added), as claimed by appellant.

Moreover, with respect to Claim 30, the Examiner has relied on the following excerpt from Kadambi to make a prior art showing of appellant's claimed technique "wherein a memory unit is connected to said first data processing unit and to said second data processing unit, said memory unit adapted to temporarily store packets before processing by said second data processing unit."

> "FIG. 2 illustrates a more detailed block diagram of the functional elements of SOC 10. As evident from FIG. 2 and as noted above, SOC 10 includes a plurality of modular systems on-chip, with each modular system, although being on the same chip, being functionally separate from the other modular systems. Therefore, each module can efficiently operate in parallel with other modules, and this configuration enables a significant amount of freedom in updating and re-engineering SOC 10." (Col. 5, lines 46-54 - emphasis added)

Appellant respectfully asserts that the excerpt from Kadambi relied upon by the Examiner teaches that "SOC 10 includes a plurality of modular systems on-chip" and that "each module can efficiently operate in parallel with other modules" (emphasis added). However, the mere disclosure of a SOC including a plurality of modular systems that can operate in parallel with other modules, as in Kadambi, simply fails to meet appellant's claimed technique "wherein a memory unit is connected to said first data processing unit and to said second data processing unit, said memory unit adapted to temporarily store packets before processing by said second data processing unit" (emphasis added), as claimed by appellant.

In the Examiner's Answer mailed 04/29/2008, the Examiner has argued that "column 5, lines 46-54, of Kadambi, teaches a plurality of systems and column 30, lines 54-66 teaches the use of priority queues that temporarily store packets." Additionally, the Examiner has argued that "column 10, lines 5-17, of Irwin discloses a data packet stored in a packet buffer before processing by the second data processing unit." Furthermore, the Examiner has stated that "[t]his limitation is clearly shown in the combination of arts."

Appellant respectfully disagrees and asserts that the excerpts from Kadambi relied upon by the Examiner merely teach that "SOC 10 includes a plurality of modular systems on-chip," where "each module can efficiently operate in parallel with other modules" (Col. 5, lines 47-54). Further, the excerpts from Kadambi teach that "[f]ilters are used for packet classification" and "[v]arious actions are taken based upon the packet classification, including packet discard, sending of the packet to the CPU, sending of the packet to other ports, sending the packet on certain COS [(class of service)] priority queues, and changing the type of service (TOS) precedence" (Col. 30, lines 59-66 – emphasis added). Further, the excerpt, in addition to Fig. 21, from Irwin teaches that "[a]s each data packet is received, the master node stores the data packet in a packet buffer (S02), and assigns a program counter to the data packet (S04)" (Col. 10, lines 5-7 – emphasis added).

However, teaching that the data packet is stored in a packet buffer as each packet is received, as in Irwin, in addition teaching that a SOC includes a plurality of modular systems on-chip, and that based upon the packet classification, various actions are taken including sending the packet on certain class of service priority queues, as in Kadambi, simply fails to meet appellant's claimed technique "wherein a memory unit is connected to said first data processing unit and to said second data processing unit, said memory unit adapted to temporarily store packets **before processing** by said second data processing unit" (emphasis added), as claimed by appellant. Clearly, storing a data packet in a packet buffer as it is received, as in Irwin, in addition to sending the packet on certain class of service priority queues based upon the packet classification, as in Kadambi, simply fails to even suggest a technique "wherein a memory unit is connected to said first data processing unit and to said second data processing unit" where "said memory unit adapted to temporarily store packets **before processing** by said second data processing unit" (emphasis added), as claimed by appellant.

In addition, with respect to Claim 30, the Examiner has relied upon Col. 31, lines 35-45 and Col. 35, lines 24-65 from Kadambi to make a prior art showing of appellant's claimed technique "wherein the apparatus includes a control logic unit coupled to an input and the policy condition table for feeding an arithmetic logic unit, which is in turn coupled to the policy action table and the state block for generating an output."

Appellant respectfully asserts that the excerpts from Kadambi relied upon by the Examiner teach that "FFP 141 includes filtering logic 1411, ...which selectively parses predetermined fields from the incoming data packets, thereby effectively obtaining the values of the desired fields from the MAC, IP, TCP, and UDP headers" (Col. 31, lines 41-45 – emphasis added). Further, the excerpts teach that "FFP 141 is shown to include filter database 1410 containing filter masks therein" (Col. 35, lines 46-47 – emphasis added). However, a FFP including filtering logic that selectively parses predetermined fields from incoming data packets, and a filter database containing filter masks, as in Kadambi, simply fails to suggest a technique "wherein the apparatus includes a control logic unit coupled to an input and the policy condition table for feeding an arithmetic logic unit, which is in turn coupled to the policy action table and the state block for generating an output" (emphasis added), as claimed by appellant.

Additionally, the excerpts from Kadambi teach that "logic 1411 in FFP 141 determines and takes the appropriate action," and that "[t]he filtering logic can discard the packet, send the packet to the CPU 52, modify the packet header or IP header, and recalculate any IP checksum fields or takes other appropriate action with respect to the headers" (Col. 35, lines 59-64 – emphasis added). However, the mere disclosure of filtering logic that can determine and take the appropriate action such as discarding the packet, sending the packet to the CPU, modifying the packet header, and recalculating any IP checksum fields, as in Kadambi, simply fails to suggest a technique "wherein the apparatus includes a control logic unit coupled to an input and the policy condition table for feeding an arithmetic logic unit, which is in turn coupled to the policy action table and the state block for generating an output" (emphasis added), as claimed by appellant.

In the Examiner's Answer mailed 04/29/2008, the Examiner has argued that "column 31, lines 20-34, of Kadambi, teaches a rules engine attached to FFP; column 35, lines 57-64, teaches filtering logic in a rules table; and column 31, lines 20-34, teach a FFP which is eventually a state machine for generating an output." Furthermore, the Examiner has argued that "[t]hese sections of Kadambi combined with the main reference of Irwin disclose this limitation."

Appellant respectfully disagrees and asserts that the excerpts from Kadambi relied upon by the Examiner merely teach that "FFP [(fast filtering processor)] 141 is essentially a state machine

driven programmable rules engine" and "[t]he filters used by the FFP... are applied on an incoming packet" (Col. 31, lines 20-23 – emphasis added). Further, the excerpts teach that "after... FFP 141 applies appropriate configured filters **and** results are obtained from the appropriate rules table 22, logic 1411 in FFP 141 determines and takes the appropriate action" (Col. 35, lines 57-60 – emphasis added).

However, the mere disclosure of a fast filtering processor (FFP) that is a state machine driven programmable rules engine, and that the FFP applies configured filters, obtains results from the rules table, and determines and takes the action, as in Kadambi, simply fails to suggest a technique "wherein the apparatus includes a control logic unit coupled to an input and the policy condition table for feeding an arithmetic logic unit, which is in turn coupled to the policy action table and the state block for generating an output" (emphasis added), as claimed by appellant. Clearly, a fast filtering processor that is a state machine driven programmable rules engine, a fast filtering processor that obtains results from a rules table, and a fast filtering processor that determines and takes action, as in Kadambi, simply fails to even suggest "an arithmetic logic unit, which is in turn coupled to the policy action table and the state block for generating an output" (emphasis added), as claimed by appellant.

Again, appellant respectfully asserts that at least the third element of the *prima facie* case of obviousness has not been met, since the prior art excerpts, as relied upon by the Examiner, fail to teach or suggest all of the claim limitations, as noted above.

Issue #3:

The Examiner has rejected Claims 17, and 18 under 35 U.S.C. 103(a) as being unpatentable over Kadambi et al. (U.S. Patent No. 6,850,521), in view of Scales (U.S. Patent No. 5,761,729).

*Group #1: Claims 17 and 18*

Appellant respectfully asserts that such claims are not met by the prior art for the reasons argued with respect to Issue #2, Group #1.

Again, appellant respectfully asserts that at least the third element of the *prima facie* case of obviousness has not been met, since the prior art excerpts, as relied upon by the Examiner, fail to teach or suggest all of the claim limitations, as noted above.

In view of the remarks set forth hereinabove, all of the independent claims are deemed allowable, along with any claims depending therefrom.

In the event a telephone conversation would expedite the prosecution of this application, the Examiner may reach the undersigned at (408) 971-2573. For payment of any additional fees due in connection with the filing of this paper, the Commissioner is authorized to charge such fees to Deposit Account No. 50-1351 (Order No. NA11P069).

Respectfully submitted,

By: /KEVINZILKA/ Date: June 30, 2008
Kevin J. Zilka
Reg. No. 41,429

Zilka-Kotab, P.C.
P.O. Box 721120
San Jose, California 95172-1120
Telephone: (408) 971-2573
Facsimile: (408) 971-4660